Project 2: Generating Images Using GANs

CSCI 297: Generative AI

Due: February 24th, 2024, at 11:59PM

Overview

This assignment focuses on Generative Adversarial Networks (GANs) to generate images of clothing items using PyTorch. The core objective is to demystify the complexities of GANs and provide hands-on experience in implementing and training these models to produce high-quality, diverse images of fashion items.

The assignment is comprised of two main components: the generator and the discriminator. You will design the generator to transform random noise into images resembling the Fashion-MNIST dataset, while the discriminator will learn to distinguish between real and generated images. This adversarial process is at the heart of GAN training and is central to understanding how generative models learn.

Through the iterative training process, you will gain insights into the dynamics of GAN training, including the balance between the generator and discriminator, and the importance of loss functions and optimizers in shaping the model's performance. You should emphasize the visualization of the data you train from and generate, allowing you to see the evolution of generated images over time and understand the metrics for evaluating GAN performance.

You will turn in a comprehensive Jupyter notebook and a report, will not only demonstrate the technical proficiency in implementing GANs but also encourage critical thinking about model architecture choices, training stability, and the potential applications and ethical considerations of generative models.

Implementation Steps

1. Dataset Preparation
   - Download and load the Fashion-MNIST dataset using `torchvision.datasets.FashionMNIST`.
   - Normalize the images to have values between -1 and 1, as this range typically works well with the ReLU activation function used in the generator's layers.
   - Create DataLoader objects for the training dataset to iterate over batches.

2. GAN Architecture
   - Design the generator model to take a latent space vector (random noise) as input and produce an image of the same dimension as the Fashion-MNIST dataset images.
   - Design the discriminator model to take an image as input and output a single scalar representing the probability that the input image is real (and not generated).

3. Loss and Optimizer
   - Use the Binary Cross Entropy loss function for both the generator and discriminator.
   - Choose an appropriate optimizer (like Adam) with suitable learning rates for both models.

4. Training Loop
   - For each epoch, iterate over the DataLoader of the training dataset.
   - For each batch, perform the following steps:
   - Maximize the log probability of correctly classifying both real and generated images.
   - Update the generator to minimize the log probability of the discriminator being correct (i.e., encourage it to generate better fakes).
   - Save model checkpoints at regular intervals.

5. Visualization
   - At the end of each epoch, generate a set of images from fixed input noise to monitor the progress of the generator.
   - Plot loss curves for both the generator and discriminator to understand the training dynamics.

6. Evaluation
   - Quality of the generated images: Are the generated images recognizable as clothing items?
   - Diversity of the generated images: Does the model generate a variety of clothing items?
   - Training stability: How stable are the loss curves? Are there signs of mode collapse?

Turn In

- A Jupyter notebook containing the implemented PyTorch code for the GANs, along with inline comments explaining the steps.
- A report discussing the model architecture, training process, encountered challenges, and how they were addressed.
- Visualizations of generated images and loss curves.